Overview

Join algorithms

- (Block) Nested Loop Join - two-level for-loop
- Hash Join - compute a hash table of one input; probe the hash table with the other input
- Sort-Merge Join - sort both tables on one of the join conditions, then merge sorted lists

Indexes

- B-Tree index - supports point and range lookup.
- Hashtable - supports point lookup.

Clustered indexes are the way the data is stored in the “original” table; they are almost always B-trees. Unclustered indexes define secondary tables that reference the main table via pointers.

Cardinality estimation - the problem of estimating the number of tuples after an operation, such as a selection or join. Good estimates are critical to cost modeling; the larger the cardinality; the larger the cost. Here we use assumptions: that the values of a table are uniformly distributed among its distinct values, and that all joins are foreign key-primary key joins.

Cost Modeling

- B(R) - the number of blocks used to store the relation R on disk
- T(R) - the number of tuples in R (also known as R's cardinality)
- V(R, a) - the number of unique values of attribute a in relation R
- M - the number of pages that fit in memory

Cost-based Query Optimization compares plans by computing their estimated cost, then chooses the one with the cheapest estimated cost to execute.

Query execution - we learned about the iterator method (iterator interface) of executing the operators in a query plan. You may see it called the "pull-based model of query execution", because each operator "pulls" data from its child operators by calling next(). The three methods used are open(), next(), and close().

Formula guide for cardinality estimation

In cost estimation, we assume data is uniformly distributed such that each distinct value has the same number of tuples.

Selectivity factor (X), assuming table R(a, b) cartesian joined S(a,c) and constants x, x1, x2:

- R.a = x => $X \cong \frac{1}{V(R,a)}$
- R.a < x => $X \cong \frac{x - min(R.a)}{max(R.a) - min(R.a)}$
- R.a > x => $X \cong \frac{max(R.a) - x}{max(R.a) - min(R.a)}$
- x1 < R.a < x2 => $X \cong \frac{x2 - x1}{max(R.a) - min(R.a)}$
- R.a = S.a (equijoin) => $X \cong \frac{1}{max(V(R,a),V(S,a))}$
- cond1 AND cond2 => $X = X_1 * X_2$

On deriving the selectivity of an equijoin:

Why R.a = S.a (equijoin) => $X \cong \frac{1}{max(V(R,a),V(S,a))}$?

Let say x0 a value such that R.a = S.a = x0, that means when we do selection R.a = x0 AND S.a = 0, the selectivity is:

$$X \cong \frac{1}{V(R,a)*V(S,a)}$$

But there can be as many as min(V(R,a), V(S,a)) distinct values of x0 (for example R has 100 value of a, S has 1000 value of a, the number of value of a after join is 100 because 100 < 1000, other S.a is filtered out. That means there can be 100 value of x0 such that R.a = S.a = x0)

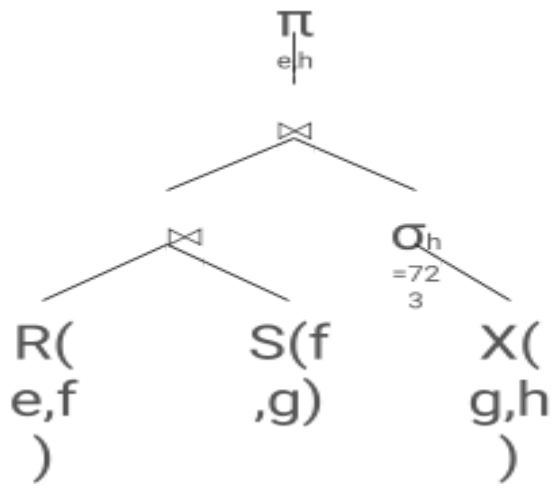
Therefore, we multiply the above selectivity by min(V(R,a), V(S,a)) which means the min value is crossed out of the denominator, leaving the maximum value. Thus

$$X \cong \frac{1}{max(V(R,a),V(S,a))}$$

Note: this is the selectivity factor. To estimate the number of tuples in a join, multiply by the T_1T_2, the number of tuples in a Cartesian product.

Problems

1. (Adapted from 414 SP 17 Final)



Consider the relations R(e, f), S(f, g), and X(g, h) in the query plan depicted above.
- Joins are natural joins.
- Every attribute is integer-valued.
- Assume that **every intermediate result is materialized** (i.e., written to disk).
- Assume that we are executing queries on a machine that has **11 memory pages** available.
- Assume uniform distributions on the attributes for the purpose of computing estimates.

Consider the following statistics:

Table	#tuples	#blocks
R	1,000	100
S	5,000	200
X	100,000	10,000

Attribute	# distinct values	Minimum	Maximum
R.f	100	1	1,000
S.f	1,000	1	2,000
S.g	5,000	1	2,000
X.g	1,000	1	10,000
X.h	1,000	1	500,000

A. Estimate the number of tuples and blocks in the selection $\sigma_{h=723}(X)$.
Select * from x where h = 723

$T(\sigma_{h=723}(X)) \approx T(X) / V(X, h) = 100{,}000 / 1{,}000 = 100$
$B(\sigma_{h=723}(X)) \approx B(X) / V(X, h) = 10{,}000 / 1{,}000 = 10$

B. Estimate the number of tuples in the join $R \bowtie S$.

This is a HARD question. Cardinality estimation of joins has been an active research topic for many years. We don't have enough statistics on R and S to make a good estimate. Here is a common estimate:

$\approx (T(R) * T(S)) / \max\{V(R, f), V(S, f)\} = (1{,}000 * 5{,}000) / \max\{100,1000\} = 5{,}000$

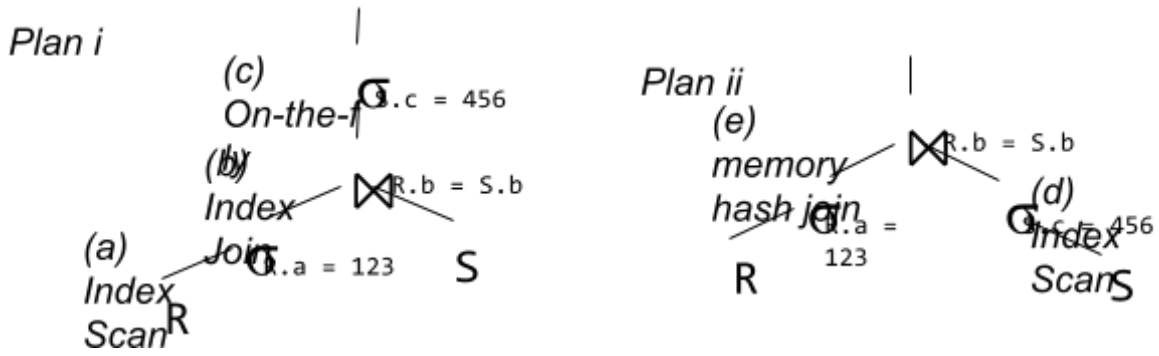
2. (CSE344 Au19 finals) Assume you have two relations, R(a,b) and S(b,c) with the following statistics:

B(R) = 4000 T(R) = 6×10^5 V(R,A) = 2×10^3 V(R,B) = 5×10^4	B(S) = 9×10^4 T(S) = 3×10^6 V(S,B) = 3×10^4 V(S,C) = 10^3

Additionally,

- Assume that indexes and intermediate results fit in memory, using pipelined execution.
- There are clustered indexes on R.a and S.b
- There are unclustered indexes on R.b and S.c
- Assume that 123 is a value of R.a and 456 is a value of S.c; that the values are evenly distributed

Two equivalent physical query plans are drawn below. Calculate the estimated IO cost of each operation lettered in the plans. Your answers should be integers, but show work for credit.



a) (2 points) Clustered index scan, Cost = X*B(R) = 4000 / 2×10^3 = 2
Cardinality, T(a) = X*T(R) = 6×10^5 / 2×10^3 = 300)

b) (3 points) Clustered index join, Cost = B(a) + T(a)*X*B(S) = 0 + 300 * 9×10^4 / 3×10^4 = 900

c) (2 points) On-the-fly, cost = 0

d) (2 points) Unclustered scan, cost = X*T(S) = 3×1000000 / 1000 =

3000

e) (3 points) In memory join, pipelined, cost = 0

f) (3 points) What is the total IO cost of each plan? Which plan would you choose to minimize the total IO cost?

plan i cost = a + b + c = 2 + 900 + 0 = 902.
plan ii cost = a + d + e = 2 + 3000 + 0 = 3002. Pick plan i since lower cost.